Universidad Carlos III de Madrid

e-Archivo Institutional Repository

This is a postprint version of the following published document:

González, R., Jiang, L., Ahmed, M., Marciel, M., Cuevas, R., Metwalley, H., Niccolini, S. (2017). The cookie recipe: untangling the use of cookies in the wild. *2017 Network Traffic Measurement and Analysis Conference (TMA), 21-23 June 2017, Dublin, Ireland. [Proceedings],* p. 1-9.

DOI: https://doi.org/10.23919/TMA.2017.8002896

© 2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

The cookie recipe: Untangling the use of cookies in the wild

Roberto Gonzalez*, Lili Jiang†, Mohamed Ahmed*, Miriam Marciel*,
Ruben Cuevas‡, Hassan Metwalley§, Saverio Niccolini*

*NEC Labs. Europe, †Umeå University, ‡Universidad Carlos III de Madrid, §Politecnico di Torino
{first.last}@neclab.eu, lili.jiang@cs.umu.se, rcuevas@it.uc3m.es, hassan.metwalley@polito.it

***Abstract*—Users online are commonly tracked using HTTP cookies when browsing on the web. To protect their privacy, users tend to use simple tools to block the activity of HTTP cookies. However, the "block all" design of tools breaks critical web services or severely limits the online advertising ecosystem. Therefore, to ease this tension, a more nuanced strategy that discerns better the intended functionality of the HTTP cookies users encounter is required.**

We present the first large-scale study of the use of HTTP cookies in the wild using network traces containing more than 5.6 billion HTTP requests from real users for a period of two and a half months. We first present a statistical analysis of how cookies are used. We then analyze the structure of cookies and observe that; HTTP cookies are significantly more sophisticated than the `name=`*value* defined by the standard and assumed by researchers and developers. Based on our findings we present an algorithm that is able to extract the information included in 86% of the cookies in our dataset with an accuracy of 91.7%.

Finally, we discuss the implications of our findings and provide solutions that can be used to improve the most promising privacy preserving tools.

I. Introduction

HTTP Cookies (cookies for short) are critical to the function of the most common and basic web services including; web mail, social networks, online retail, advertising, etc. They are also the most widely used method to track the activity of users online [1] and may be used to transfer personally identifiable information (PII) between a user's browser and online services [2].

Recent studies consistently point to the growing disconnect between users' desire for privacy and the online services ecosystem's desire for more/higher-quality information about users [3]. In particular 28% of the US citizens admit they have used the Internet in ways to avoid being observed or seen by advertisers [4]. In response to users' desire for better privacy controls, there is today a burgeoning ecosystem of tools, services, and business models that aim to help users to better manage their privacy.[1] Consequently the deployment of anti-tracking tools by users is reported to be growing [5]. In response, we have also witnessed the appearance of more aggressive techniques to track users browsing patterns that make use of HTTP Cookies. For example, evercookies [2] are used to avoid users deleting certain cookies and cookie matching/syncing [6][7][8] is used to aggregate the knowledge different players have about a user.

Today users can protect their privacy by blocking third party cookies and/or inserting the (often useless [9]) *Do Not Track* (DNT) header, both methods are available in most of the modern browsers. Users can also deploy third party tools such as browser plugins and network proxies. These tools (including NoScript, Adblock Plus, Ghostery, and Disconnect) give users a sense of security, however, they present serious drawbacks, such as blocking from 6% to 21% of the functional scripts deployed in web pages, and allowing the execution of between 37% and 78% of the tracking scripts [23].

First, by virtue of their design, these tools typically break some of the legitimate functionality of websites. Second, they cannot protect users from tracking cookies set via first parties [1]. Due to the complex relations that first and third-parties establish [1][5][11], blocking cookies at a domain-level alone does not protect users [1]. Lastly, it was recently estimated by PageFair and Adobe [12] that 6% (198M) monthly active users deployed ad-blocking in Q2 2015, resulting an estimated loss in revenue of over $21B. Therefore, web services have started a fight against ad blockers [13], and it is now common to find many popular services that require users to deactivate the protection tools in order to access content.[2]

All these well known problems have encouraged the researchers and developers to try to define fine-grained solutions to protect the privacy of users with a minimal effect in the online advertising ecosystem, c.f. [14][15][16][17]. Nevertheless, while these proposals are promising, they have been designed without a comprehensive understanding of the actual information sent within the cookies. This fact could make them either to break some of the web-services they intend to fix, or to provide a false sense of security to their users.

In order to improve existent tools and develop new tools that can protect the users without side effects a study of the real use of the cookies is needed.

This paper presents the largest measurement study to date on the utilization of HTTP cookies in the wild. Using a dataset composed by more than 5.6B HTTP connections, from thousands of real users in a European metropolitan network,

[1] see Adblock Plus, Ghostery, Privacy Badger, Blur, Disconnect and etc.

[2] c.f. http://www.wired.com/how-wired-is-going-to-handle-ad-blocking/

we study the utilization, structure and content of HTTP cookies. First, we look at the schematic structure of cookies and find that they are more complex than what is assumed in both the academic literature, and by the multitude of tools available to help protect the privacy of users. We find that cookies routinely obfuscate their contents by using a variety of proprietary formats. Moreover, because current studies do not look into the cookies in any great detail, we find that they make naive assumptions regarding how cookies are structured and utilized. In turn these assumptions lead to a gross over simplification of the complexity of cookies, which results in users having a false sense of security. Our findings reveal that at least 60% of the cookies we find in the wild contain complex information that cannot be interpreted in simple ways. Inspecting the contents of cookies, we develop a novel method to extract fine-grain information from the data carried in these complex cookies. Last, we apply these methods to the cookies in our dataset and provide analysis of their contents.

Contributions

- We analyze cookies sent by thousands of real users, in contrast to the vast majority of previous studies that based their results on traces generated using active measurements[1][3][15][18][19][20]. The results of our analysis show that the cookie `name` and `value` fields are routinely packed with multiple values, which can change frequently. This makes it difficult for fine-grain tools [14][16] to correctly interpret the behavior of cookies.
- We evaluate some of the common assumptions regarding the identifiability of tracking cookies, finding some of them to be wrong. Common assumptions, such as "they are long lasting", "have high entropy" and are "constant and unique for every user" [15][20] are found not to hold in a multitude of cases. This in turn breaks the basic assumptions made by state-of-the-art methods such as [14][15] on the identifiability of 'user identifiers'.
- We develop a method to unpack and parse the contents of cookies with high accuracy. Our simple and scalable method is able to parse 86.2% of the cookies with an accuracy of 91.7%. The formats identified reveal cookies sending PII in clear text or hiding the user identifiers in complex data structures. The structures obtained with our method could be used directly to improve the results of some previous works.

II. Background

Our analysis is based on passive measurements collected from a 10Gbps link (Endace DAG 9.2X2 capture card), serving several thousand users, across several academic institutions, at a Large European metropolitan region. We collect from the TCP/IP headers the (*src/dst IP, src port, seqn*), and *only* the HTTP request headers, on port 80. We analyze in total approximately twelve weeks of traffic (19/01-9/04 2015) containing more than 40Tb of network traces. We generate 3.8Tb of metadata, comprising of 135K IP addresses, that make 1.6 billions TCP connections to 3.5M different hosts, and send 5.6 billion HTTP requests.

A. Ethic Considerations

The data collection and experiments presented in this paper have been conducted in the context of the EU-H2020 project TYPES. UC3M and NEC have obtained the approval for the research activity conducted in the context of such project from its IRB committee and Data Protection Officer[3], respectively. Moreover, UC3M obtained the consent of the Network Operator to collect the data and use it for research purposes.

To the best of the authors knowledge, the data collection, storage and processing conducted followed the European as well as Spanish and German Data Protection regulation. In particular, we collect exclusively HTTP request and only analyze the protocol header. Once the metadata is obtained the raw data is deleted and only the metadata is stored outside the measurement server. In the data processing the source IP address is anonymized using one-way hash functions.

B. HTTP Cookies

HTTP cookies are the standard mechanism used to track state in the browser of users as defined by among others the IETF RFC6265 [21]. Briefly, a cookie is defined by the tuple; {`Name`=*Value*; [`expires`=*Expiration_Date*;] [`path`=*Path*;] [`domain`=*Domain_Name*;] [*secure*]}. To set a cookie, a host either adds the *Set-Cookie* header in HTTP reply messages to user requests [21], or writes an entry via JavaScript. Once a cookie is stored at a user's browser, if not explicitly blocked, the browser adds the cookie's `Name=Value` tuple, to the header of each HTTP request to assets from hosts from the `domain` defined in *Domain_Name* and in *Path*, until the *Expiration_Date*. To isolate resources between domains, the *same-origin* policy is used. It states that domains may only set, read, and modify cookies which match their *Domain_Name*, and resource *Path*.

1) First and third party cookies: When a user visits some web page W_1, W_1 can in turn initiate connections to different domains $W_2 ... W_n$ (i.e., to download images from a CDN, ads from an ad-network, etc.). We refer to the cookies set by the domain the user intend to visit (W_1) as first party cookies and any of the cookies set by $W_2 ... W_n$ as third party cookies. Moreover, third party cookies can be used for operational purposes and some Internet services break if they are blocked. In contrast, they can also track users online.

The same cookie can act as both third party cookies (i.e., Facebook cookies when you see a widget in other website) and first party cookies (i.e., the same Facebook cookies when you visit facebook.com). Therefore in this work, we do not differentiate between first and third party cookies. Nevertheless, the common way to track users with cookies needs them to act as third party cookie.

C. Cookie based tracking

The most simplistic scenario for cookie based tracking occurs when a user U_1 visits web page W_1, the browser can make

[3]Note that L. Jiang and H. Metwalley contribution to this work was developed while they had a contractual relationship with NEC Labs Europe and thus the approval from NEC Data protection Officer was valid for them.

HTTP requests to one or more trackers ($T_1 ... T_n$) adding its URL in the *Referer* header of the HTTP protocol. In this way, T_1 can set in the users browser a cookie C_{1id} and know that the user with that cookie has visited W_1. Moreover, when the user visits a different web page W_2 that also contains T_1 it will send the cookie C_{1id} together with the *Referer* header of W_2. In this way, T_1 has the knowledge of two different sites (W_1 and W_2) visited by U_1. This continuous process is used to track the activity of users, and if the tracker is present on large number of domains it is able to collect a significant quantity of data regarding the browsing habits of a user.

III. Inside the cookie

While the reach of the cookies gives us an intuition on the amount of information a potential tracker can infer about the user, it does not inform about the information that is contained in the cookie itself.

This section analyzes the information included inside the cookie. As shown in section II-B, a cookie is defined by a *Host* (i.e., google.com) and a *Cookie Name* (i.e., `NID`) and when used as a third party it can track the browsing history of users among different webpages. Moreover, it contains a *Value* that is assigned to that cookie name. To build an intuition of the scale of the tracking ecosystem networks, in the following we study the reach of different trackers. Then, we respectively analyze the information inside the *Cookie Name* and the *Cookie Value*.

A. Cookie reach

With respect to domains, the reach of a cookie is the number of different referrers seen in HTTP connections sending the cookie. For example, tracking cookies are referred by many different domains, while functional cookies only appear with the domains that utilize them. Understanding the cookie reach based on the number of different domains is of key importance to understanding the tracking ecosystem - because it enables us to quantify the fraction of user activity across domains that a given cookie is able to observe.

Figure 1 gives the complementary cumulative distribution function (CCDF) of the number of different domains included in the *Referer* header of HTTP requests when a given cookie name is sent by a user. We observe that vast majority of the cookies have a small reach indicating that they play a functional role for the sites that set them. In particular, 99.24% of the observed cookies have the ability to track $\leq$5 domains. However, we find cookies such as the `id` of doubleclick.net that is sent together with almost 162K different domains in the `Referer` header. Moreover, we find 212 cookies that could potentially track more than 10K different domains. Among these cookies, we observe the vast majority are directly related with online advertising ecosystem, for example, the aforementioned `id` cookie of doubleclick.net, the `icu` of adnxs.com or the `uid` of criteo.com that reach 162K, 46K, 21K domains respectively. It is also found that cookies operated by services that are not directly related with the advertising (even when clearly track users online), such as the `uid` cookie of addthis.com and the `__stid` of sharethis.com with a reach of 65K and 21K

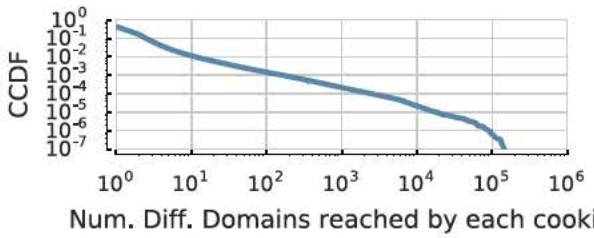

Fig. 1: Most of the cookies can see a single domain (normal webpage cookies), nevertheless, a long tail of cookies can control a huge number of domains. doubleclick.net cookie *id* has been referred for almost 162K domains.

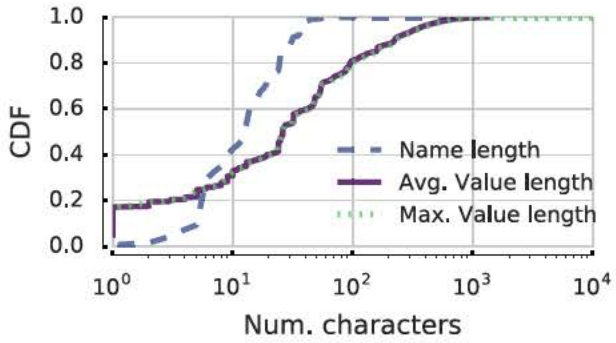

Fig. 2: Only 1% of the cookie names have more than 45 characters. Nevertheless, some names have hundreds of characters. Moreover, for the cookie values, the distribution of the Avg. length is similar to the distribution of the max length, it indicates that most of the cookie values have a fixed length.

domains respectively, and the `UID` of scorecardresearch.com which reaches more than 31K domains. These cookies, while not directly related with the distribution of online ads, still track the users and directly obtain information that can be utilized for individual user profiling.

B. Cookie name

The *Cookie Name* is typically assumed to be intended to act as a variable name; providing developers with a reference to access the information inside the *Cookie Value*. It is normal to define the cookie itself with its name and as such it has been treated in this way by previous works [20][14][15]. In fact all the previous work assumes that websites use a fixed set of cookie names independently of the user, thus, they draw a conclusion that the *Cookie Name* is not expected to contain any relevant information about the user. This section aims to understand whether this assumption holds or if additional considerations are needed.

To test this assumption, we first look at the typical character length of the cookie name. Figure 2 shows the CDF of the number of characters per *cookie name*. 50% of cookies have a length smaller than 13 characters, a number that seems reasonable for a variable name. Moreover, more than 99% of cookie names are composed by less than 45 characters. However, we also find some names composed by hundreds of

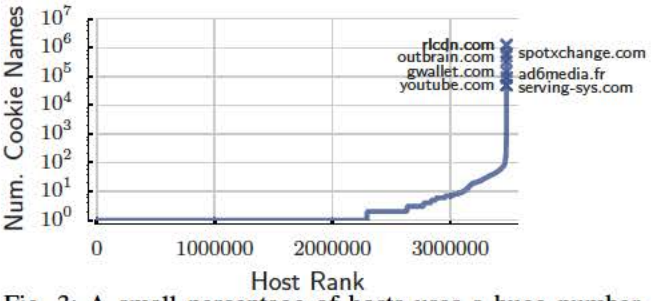

Fig. 3: A small percentage of hosts uses a huge number of cookie names because they tailor information into them.

characters. For example, krxd.net; a domain belonging to the data management company Krux[4] , uses the cookie name to store different values separated by the pipe symbol "|":

```
e_[id]^adid|[Number]^creativeid|[Number]^siteid|
[Number]^campaignid|[Number]^placementid|[Number]
=[timestamp]
```

This kind of cookie names can grow and store a large quantify of information invalidating common assumptions that treat the *cookie name* as a simple reference to the *cookie value*.

Furthermore, the tailoring of the cookie name to add values means that these domains use a large number of cookie names, for example, more than 22K different cookie names are contained in our dataset for krxd.net. In turn, this makes it difficult to block/tamper with specific cookies using the cookie name. In particular, previous solutions that aim to automatically find unique identifiers [14] would not work with these cookies. Moreover, other solutions intended to block or modify cookies [16] would find difficulties to adapt their behavior and handle these special cases.

In the following, we analyze those domains using a large number of cookie names. Figure 3 presents the number of cookie names used per host. Usually a small number of cookie names is used per hosts, nevertheless, there exists a small number of domains that make use of thousands of cookie names. Unsurprisingly, most of these domains are dedicated to the online tracking or online advertising, with rlcdn.com (a domain belonging to liveRamp.com) using more than 1.12M cookie names. This domain seems to add the user identifier inside the cookie name, with cookie names starting with `drtn` or `dids` followed by an id of the user. Not all domains in the top list are adding user ids to their cookies. Some domains use a large amount of cookie names for operational purposes, without "hiding" user identifiers. For example youtube.com uses cookies with the format $[VIDEO\_ID].resume = [time]$ to save the time when the user stops watching a video, allowing the user to resume the view, thus, youtube.com is potentially using a number of cookie names equal to the number of videos in their system.

We leave the methodological analysis of the specific information contained inside the cookie names as future work.

[4]http://www krux.com/

C. Cookie value

The cookie value is the part of the cookie originally designed to store the information. We therefore try to untangle the quantity and type of information contained in it. In particular, we are concerned with two features used in previous works to recognize unique identifiers [20]: the length of the *value* (stable, constant length, passes the entropy test) and its stability to finally identify exactly the type of value inside it.

1) Value length: The length of the value defines its potential entropy. Figure 2 shows the CDF of the average and maximum length (in characters) of the cookie values. First, 1% of the cookies send empty values and 17% of them have always a single character. Most of the cookies sending an empty value do it for functional purposes. For example, several domains set an empty cookie named `__test` to check whether the user browser accept cookies or not. For cookies for which the value is always composed by one character the usage varies, such as the `dnt` cookie used by some trackers to indicate the user has opt-out of the tracking.

Very small values cannot hold a user identifier since the amount of entropy is small and they would not be able to correctly represent a large number of users. Nevertheless, 72% of cookies present average value length longer than 7 characters and could potentially be used as unique identifiers. Furthermore, the distribution of average and maximum value length is almost the same for all the values. It only presents differences in the tail. This similarity indicates that most of the cookie values have a fixed length. Indeed, 65% of the cookies in our dataset use always the same length in their value. This behavior is consistent with the expected one for an identifier in previous works but also with other functional cookies (e.g., a cookie storing language setting of the user, ISO 639-1 format).

On the other side, for some cookies the maximum size differs significantly from the average size. For these cookies the value length appears to grow. They concatenate information about the users while they browse the web. As an example the `lr_udsc` cookie is used by liverail.com to store information about the user. This specific example in our dataset adds hexadecimal values separated by the % symbol to the previous cookie value.

2) Value stability: Another typical requirement for cookie values to be used as a unique identifier is that it should be long lasting. Nevertheless, the web sites could periodically change (update or rotate) the value of their cookies while still tracking the whole browsing history of the user. To analyze this matter we study the length of time a cookie value remains the same.

Figure 4 shows the CDF of the median duration of a cookie value for all the cookies, for the cookies with values bigger than 7 characters and the 1K most used cookies in our dataset. For 69% of the cookies the median duration of the value is less than 1 day, this duration can be understood as the duration of a browsing session. Moreover, it is less than 1 week for 79.4% of the cookies. These values are even higher in the case of the cookies with enough entropy to be unique identifiers (those ones with values longer than 7 characters). Nevertheless, when we focus on the most popular cookies the scenario is

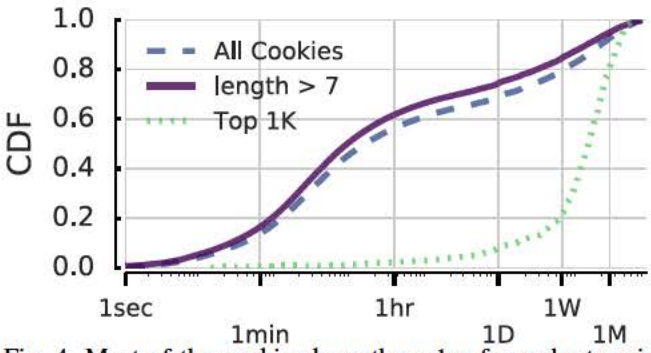

Fig. 4: Most of the cookies keep the value for a short period. Nevertheless, cookies reaching a large number of users tend to use more stable values

Type	%
Hexadecimal	18.96
Text	7.00
Number	6.53
TimeStamp	3.77
URL assign: $name1 = val1\&name2 = val2...$	1.31
JSON	1.02
True or False	0.88
Simple assign: $name = val$	0.84
URL	0.40
IP address	0.05
Email	0.04
Other (Unsure)	57.42 (1.8)

TABLE I: Percentage of cookies with each kind of type.

completely different, and 782 of the top 1K keep the same value of the cookie for more than 1 week.

The results suggest the most common kind of cookie is associated to a session. Nevertheless, the most popular cookies keep their value constant for long time being able to act as identifiers.

3) Value type: In this subsection we look at inferring the kind of data that is embedded inside the cookies. A first inspection indicates the existence of completely different data types as well as complex formats inside the cookie values. Moreover, we found that around 20% of the cookies are encoded using Base64 and/or URL encoding technologies to avoid problems with possible non-standard characters.

In order to understand the information contained inside the values we first apply iterative decoding, and then, we extract the value type using regular expressions.

a) Iterative Decoding: The first try to decode the cookies in our dataset reports an unexpected case: after decoding some of the cookies, they seemed to be still encoded. While we have not been able to understand why webpages encode the information more than once, we have found some of them that encode their cookies several times. To correctly decode these cookies (and all the others). For every cookie value, we first check if it is base64 or URL encoded string and decode them. If the process does not modify the cookie we leave the cookie value as it is. In case any of the decoders is successful, we apply the process again to the resulting value until the decoder does not provide any new result.

b) Extracting the value type: After the cookie value is decoded, we apply rule-based methods to identify some of the possible data types including simple types like numbers, hexadecimal values, or text; specific formats like email addresses or URLs; and complex types like JSON, or the format used to pass parameters in the URLs. We apply regular expressions to all the values sharing the same domain and cookie name and if we find the same value type in at least 95% of the cases we assign that type to the cookie. If, for a given cookie, we are able to identify the type for at least 95% of the values, but they are recognized as different types, we define the type as *Unsure*. We mark the rest as *Other*.

Table I shows the percentage of cookies that belong to each category. More than 40% of the cookies in our dataset belong to one of the predefined categories. The analysis of the cookie values semantics is out of the scope of this paper, however, a simple inspection discovers more than 3K hosts sending an email in clear text with their cookies. It can be considered as a leakage of PII. An inspection of the cookies sending the mail reveals most of these cookie are sent by domains using the popular CMS Wordpress that uses a cookie called `comment_author_email_[uniqueID]` where they store the email of the user.

Furthermore, almost 19% of the cookies store hexadecimal values. This type of data is the one used inside the `PHPSESSID` cookie used by thousands of different domains to keep the user session. Nevertheless, in general we cannot know if the value is somehow encrypted information or an identifier of the user. More surprising is the use of complex formats as JSON, used by more than 72K domains. This format is used to pack different pieces of information inside the same cookie, instead of using different cookies for that purpose. This kind of construction makes difficult the detection and modification of cookies in a similar way as the tailoring of cookie names reported at the beginning of this section. These cookies would force the detection systems to individually interpret the data inside every JSON field. Finally, almost 60% of cookies whose data does not belong to any of the analyzed types. A good example is the ubiquitous `__utma` cookie of Google Analytics. It stores multiple different variables inside:

```
__utma=DomainID.UserID.FistVisitTS.
       PreviousVisitTS.CurrentTS.NumSessions
```

It is worth noting that most of these 60% of cookies use proprietary formats to pack different values. As an example, the `__utma` cookie packs 6 different values including a user identifier and a timestamp of the current visit. Thus, the timestamp makes the value change every time the cookie is sent, even though the user id stays immutable. Thus, a cookie including a unique identifier does not need to be long lasting.

As in the case of the JSON format, this construction makes difficult the manipulation of the cookie. Furthermore, it does not only affect our ability to automatically modify the cookies on the fly as proposed by other works [16]. It also affects our ability to correctly store (or share) the data inside the cookies in a privacy preserving way. A simple anonymization of the value (i.e., using a hash function like MD5) would render

useless the data for any future analysis. Then, the only way to correctly process the information is by understanding the format used to store different values and handle every piece of information independently.

IV. Entity based Cookie Interpreter

Having observed that cookies may convey non trivial information (in their name and value fields), and that this information may be encoded in a complex manner, in this section we propose a method to unpack the contents of cookies in an unsupervised manner. The primary challenge to be able to correctly parse the contents of cookies is that while the standard dictates the need for a `name=value` tuple be specified, it does not place any structure or limitation on the contents of either. In turn, cookie origins are able to define arbitrarily complex schemas to encode `values` and as shown in the previous section, pack the `value` field with many variables. As an example, consider the following individual cookie value:

```
''253024271.1453122666.239.16.utmcsr=host.com/sea/|
utmccn=(10.0.0.5)|utmcmd=org|utmctr=ail@host.com}''
```

First, the cookie contains several different types of fields, specifically, a Unix timestamp ('1453122666'), an email address ('ail@host.com') and an IP address ('10.0.0.5'). Second, we observe several delimiters ('.' and 'l') that are combined to form the text. Third, delimiters may be a part of the content of fields without any special demarcation, e.g., in the example above, the character '.' is both a delimiter and content, i.e., within the IP address.

To automatically interpret such cookies, we develop an unsupervised entity-recognition method that unpacks the contents of the cookies. Our method is able to extract the individual fields in cookie values and interpret the common patterns for cookie values for different cookie names from each domain.

A. Entity definition

Similar to the concept of named entity in natural language processing[22], we define an entity in this study as a single, indivisible piece of information in the value of a cookie. Examples of entities are IP addresses or timestamps. Because the cookie developers are free to define any arbitrary field, instead of attempting to identify an arbitrary list of possible fields, we have adopted a pragmatic approach and analyzed more than 100K randomly selected cookies to identify the most common entity types. Table II lists the most common entity types (i.e., numeric values, alphanumeric text, boolean, hexadecimal, json, timestamp, email, url, and ip address).

To accurately identify entities within cookie values, based on the characteristics and compositions of different types of entities, we distinguish between singular entities and composite entities.

Singular entity: Singular entities refer to those ones that cannot contain a different entity inside. It includes *numeric values, timestamp, alphaNumeric text, boolean, hexadecimal*.

Composite entity: Composite entities aggregate numerous singular entities in an arbitrary order. It includes *json, email, url, and ip address*. Clearly, the composite entity may contain singular entities or other composite entities as part of its content.

	Entity type	Example	Symbol
Singular	Numeric	*17*	N
	AlphaNumeric	*NewYork*	A
	Boolean	*True or False*	B
	Hexadecimal	*234AD2A5*	H
	Timestamp	*142800298*	TS
Composite	JSON	*a:b, c:d...*	J
	Email	*abc@domain.com*	E
	URL	*webofknowledge.com/*	U
	IP	*201.202.1.3*	I

TABLE II: Entity types and samples

In the previous section about 40% of the cookies have been identified to be formed by a single composite or singular entity. Moreover, the remaining 60% of the cookies may be formed of more than one entity using a proprietary format. In order to avoid losing of information we have to extract different pieces of information in the correct order (e.g., if we interpret an IP address as 4 numbers separated by dots, then, we lose the chance of checking whether that 4 numbers in fact form an IP address or not).

B. Entity extraction priority

Composite entities may contain singular and/or composite entities. For instance, an email may be part of a url (e.g., https://domain.com?userEmail=myemail@mydomain.com). Based on this fact, we define an extraction order.

First, the JSON objects should be extracted because it can contain any of the other data types. Next, we extract the other composite data types, both IP addresses and email addresses may be included in an URL, thus, we will first identify the URLs, following we will extract the email and IP addresses.

Finally, the simple entities will be extracted. In this case we have to start for the more specific entities like the timestamp and the boolean that could be also identified as numeric and alphanumeric, respectively. Following, the hexadecimal values will be extracted to end up with the numeric and the most general alphanumeric ones.

C. Cookie interpretation and representation

After the entity extraction, we represent each cookie value using an entity type as presented in the following example. Here, "N" denotes numeric value, "TS" denotes timestamp, "A" denotes alphanumeric text, and "U" denotes URL. This kind of entity-type representation enables us to interpret cookies in a fine-grain way and can be used for post-processing.

```
Input:
''250607554.1422361545.2.2.utmcsr=host.com|
utmccn=(referral)|utmcmd=referral|
utmcct=thesection''

Output: N.TS.N.N.A=U|A=(A)|A=A|A=A
```

Format selection: After all the values associated with a cookie have been interpreted we obtain a set of value representations. Those value representations may be different for different values of the same cookie if the format used by the hosts changes. Thus, it may happen that the selection of

Algorithm 1 Entity-based cookie interpreter

Input: Set_type, $Set\_cookies$; β; $count$=0
▷ decoding
for $cookie$ in $Set\_cookies$ **do**
  while $count++ <= \beta$ **do**
    check *IsEncoded*($cookie$) ▷ *Base64* **or** *URL*
    if True **then**
      $cookie = decode(cookie)$
    end if
  end while
▷ entity extraction
  for E in *Set_type* **do**
    markup all entities typed with E in $cookie$ (Section IV-A)
    extract entities by extraction priority (Section IV-B)
  end for
end for
▷ *format selection*
Cookie format generation by following Section IV-C

the format used by a cookie is not straightforward. In this case we define a threshold α for the minimum percentage of values that should have the same format to define it as the general cookie format. We tuned different values finding $\alpha = 90$ as the optimum trade-off between accuracy and recall.

Moreover, some of the fields are specific cases of others. For example, a numeric value can be sometimes classified as a timestamp or an alphanumeric value could be extracted as hexadecimal. To avoid those cases, we use also a priority selection in which the most specific entity (i.e., timestamp) is selected if it is in at least $\alpha\%$ of the cases, if not, it is interpreted as a more general entity (i.e., number).

With this method we have evaluated the format used by more than 5M different cookies.

Results analysis: Contrary to the figures in Table I, only 5.7% of the composite cookies include hexadecimal values while 69.2% of them include numeric values and more than 22% include text. Moreover, 35% of the cookies pack a timestamp together with other values. Thus, one third of the cookies may include unique identifiers even when their values change continuously. Finally, more than 600 cookies pack the email of the user inside the cookie value and 8300 cookies includes an IP address, also considered as private information by the American and European data protection laws.

D. Algorithm of entity extraction

Algorithm 1 summarizes the whole process of entity-based interpreter, where Set_type is the entity type list as shown in Table II. $Set\_cookies$ contains all cookies to be processed for entity extraction. β denotes threshold for iterative decoding.

E. Validating the selected format

The entity based cookie interpreter described above allows us to identify the format for more than 5M cookies, that is, the 86.3% of all the cookies that have been sent more than 10 times[5] in our dataset. However, the total absence of a ground truth limits our possibilities to validate the obtained results.

To overcome this problem we have proceeded to do manual validation. To this end, we have randomly selected 1000 of the cookies whose formats is to be identified and we have asked 5 independent persons to manually check if the format assigned fits with the one of the cookie values. It is important to remark that in some cases it is difficult, even for human beings, to identify the underlying format (e.g., some annotators were too confused to differentiate a URL from pieces of text using the slash "/" as delimiter or even identifying if a string is a JSON or not). Thus, we gave our independent annotators 3 different options. They could select whether the cookie values fit the format, do not fit it or if the annotator is not sure. Then, we select as the correct answer that one agreed for more than 50% of the annotators sure of their answer. In case of a tie, we select the most conservative option. This is, if 2 annotators vote the format is wrong, other 2 vote that it is correct and the fifth one is not sure, we mark it as wrong. Nevertheless, these cases are residual since the 5 annotators gave the same answer in 66.6% of the cases.

The results indicate our interpreter identifies the correct format for 91.7% of the cookies. We consider this accuracy very high for the first implementation of the entity based cookie interpreter. A manual inspection indicates that our system fails to interpret three types of cookies: cookies that use more than one format depending on the user/moment and cookies that grow, adding more information and changing the format with the time. In both cases, a manual inspection of the cookie values is needed to correctly infer the format; 3) the third type of incorrectly interpreted cookies are in binary format encoded using Base64 (or other encoders). It is impossible to entangle the information stored inside them if they are only set and read by the server. Moreover, if they are set using JavaScript, only a case by case reverse engineering of the JavaScript code could shed some light over the format they use.

V. Related Work

The research community has studied online advertisement and its privacy issues. These studies have been mainly active measurements, using simulated traffic, with mostly the Alexa top-K sites [1][18][3][19][20][15]. Furthermore, Ikran et al. [23] follow a different approach and use machine learning to identify the javascript snippets used to track users online. Only the paper from Metwalley et al. [5] presents a passive measurement study related to advertisement. In [5], authors find that the top 5 trackers contacted more than 90% of the users.

Related to the study of cookies, works such as [20][15][2] study how cookies are used as identifiers by trackers. In order to protect users from this phenomenon, Papaodyssefs et al. [16] propose a proxy that maps public users cookies to private, always keeping the anonymity of the user. The problem is that

[5]If a cookie has appeared less than 10 times we cannot infer any format inside it.

they treat cookies as a whole string, when we have shown that cookies zip several values on them. Another work [17] protects users by blocking identifiable information from HTTP requests. But, according to our knowledge, this is the first work that extract information from HTTP cookies characterizing them.

Englehardt et al. [20] find that it is possible to reconstruct up to 73% of user's history using their cookies alone. Li et al. [15] find that 46% of webpages in the Alexa top 10K have a third party tracker, and 30% use at least one of the top 5 ranked trackers. Roesner et al. [1] find similar rates of tracking of users, but show also that first and third party domains commonly bypass the *same-origin* policy protections in order to track users. Moreover, [24] study the information exposed when a user cookie jar is cracked.

Recent studies [1][18][5] show that users online are tracked continuously. As well as cookies, methods include HTML5 Local Storage, Etags, Flash and Finger-printing [2][25][26]. First and third-party domains collect data on users and exchange information through a complex of web relations [3][1][19]. All with the aim of profiling the complete online activity of users [18][27], typically for targeted services.

VI. Discussion

The insights obtained from the analysis of the data report two main implications for the users. First, cookies (not surprisingly) are still used to transfer PII (Personally Identifiable Information) of the users. Second, the tools available today to protect the online privacy of user need to be greatly improved to address the complexity of modern cookies.

The existence of cookies sending PII in clear text represents a serious leakage on the users privacy. For example, any agent with access to the network data (i.e., a malicious user sniffing traffic in a public WiFi network) could easily identify the real user behind the connection if the user's email or other sensitive information is sent inside a cookie.

However, online advertisement represents one of the main source of revenue for webpages and the privacy preserving tools available nowadays (i.e., Ghostery, AdBlockPlus, NoScript, etc.) are designed to block the ads. Consequently, they severely damage the freemium Internet ecosystem. Furthermore, they also affect the user experience by breaking some legitimate services. Finally, publishers have started an arms-race against the ad-blockers that may end up with the users adding exceptions, thus, removing all the protection against privacy leakages.

A. Integration of our solution in existent tools

The research community is moving to design and develop new fine-grained privacy protection tools that allow the online advertising while still avoiding the privacy leaks[16][17]. Nevertheless, these methods make simplistic assumptions about the composition of the cookies that could be easily fixed by using the results of this work.

In particular, the methodology to detect unique identifiers in cookies of our previous paper [14] can be easily extended to include the format of the cookie value, comparing individual pieces of information instead of the whole text.

	Entity type	Anonymization Action
Singular	Numeric	*Hash(number)*
	AlphaNumeric	*Hash(text)*
	Boolean	Keep value
	Hexadecimal	*Hash(value)*
	Timestamp	value + *random noise*
Composite	JSON	Analyze each variable separately
	Email	*Hash(user)*@domain.com
	URL	host/*Hash(path)*
	IP	*Mask(IP address)*

TABLE III: Anonymization actions

Moreover, solutions like WIT [16] (a network proxy that translates the user cookies into private ones avoiding the tracking) would fail to detect the tracking with its actual model. However, as in [14], the knowledge of the format used inside the cookie value would help the tool perform more accurately. Furthermore, WIT also needs to know when cookies include the user identifiers in the cookie name as demonstrated in section III-B.

The Cliqz browser [17] already blocks all the third party cookies protecting the user privacy. Nevertheless, as the authors admit it causes some services to break. They could use the methodology described in this paper to analyze the format of third party cookies and block only those parts that make the users unique.

B. Other implications

The detection of cookie syncing has attracted the attention of the research community in the last years [7][8]. Moreover, this detection is most times based in detecting the transfer of the id used in the cookies to third parties. In order to be able to do this effectively, we need to correctly detect the user IDs. The cookie structures detected in this paper can be used as a pivotal and fundamental step towards that end.

Finally, the output of the cookie interpreter can also be used to properly anonymize datasets containing cookie data in a meaningful way. A naive approach would hash the entire cookie value in order to anonymize it. However, it would make impossible any further analysis of the information inside the cookies (i.e., to find user identifiers). We propose to anonymize each piece of information inside a cookie independently (using the structure identified by the interpreter). That way, we propose to apply different anonymization operations depending on the type of each piece of information as shown in table III. It would keep the privacy of the users, and at the same time, it would allow further analysis of the data.

C. Available datasets

In order to help developers and the research community to use the results of this paper we make public two datasets. One containing a list of domains that use to include information inside the cookie name and a list with the structure found for more than 5M different cookies.

The first dataset provided can be found in https://privacyaware.nlehd.de/data/NumCookiesPerHost_More_1K.csv and includes the name of all the hosts that use more than one thousand different

cookie names as described in Section III-B (the tail of Fig. 3). The hosts in the list most probably include some kind of information inside the cookie name. This form of cookie name tailoring may affect some of the state-of-the-art solutions for protecting the privacy of the user by intercepting the cookies. These tools can treat differently the information coming from the host in the list and researchers can use it as a starting point to better understand this practice.

Moreover, in https://privacyaware.nlehd.de/data/patternListComplete.csv we make available a dataset containing the list with more than five million cookie value structures discovered using the cookie interpreter described in Section IV. This dataset can be used to process each piece of information inside a cookie separately. It will improve the accuracy of the methods devoted to identify unique identifiers but also those ones that try to identify cookie syncing. Finally, it can be used to correctly anonymize other datasets including cookie data in a meaningful but private manner.

VII. Conclusions

This paper is the first study that analyzes how the cookies are used in the wild in a fine-grain way on a large dataset (more than 40Tb network data) coming from real users.

We discovered the structure of the cookies is much more complex than it has been assumed by previous works that following the standard defines a cookie as a pair composed by a name and a value. We find thousands of websites that tailor the cookie name to include a unique identifier of the user. It makes difficult to compare cookies among users, moreover, it affects our ability to generate black lists of cookies to block only those ones that represent a risk for the user privacy without breaking any of the Internet services.

Furthermore, we find almost 60% of the cookies do not use a simple format for its value. Instead, they pack multiple independent values inside a single cookie using proprietary formats. It makes impossible to correctly analyze the information inside without knowing the format used. Thus, we have developed a methodology that iteratively extract the different pieces of information inside the cookie values. Our methodology is able to infer the format with a precision of 91.7% and a recall of almost 80%.

Acknowledgments

We thank anonymous reviewers and our shepherd for their valuable comments and suggestions. This work has been partially supported by the European Union through the H2020 TYPES (653449) and ReCRED(653417) Projects.

References

[1] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the web," in *NSDI*, pp. 12–12, 2012.

[2] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web Never Forgets: Persistent Tracking Mechanisms in the Wild," in *ACM SIGSAC*, pp. 674–689, 2014.

[3] B. Krishnamurthy, K. Naryshkin, and C. E. Wills, "Privacy leakage vs. protection measures: the growing disconnect," in *W2SP*, 2011.

[4] "The state of privacy in post-Snowden America." http://www.pewresearch.org/fact-tank/2016/09/21/the-state-of-privacy-in-america/, 2016.

[5] H. Metwalley, S. Traverso, M. Mellia, S. Miskovic, and M. Baldi, "The Online Tracking Horde: A View from Passive Measurements," in *TMA*, pp. 111–125, 2015.

[6] "Cookie Matching." https://developers.google.com/ad-exchange/rtb/cookie-guide, 2016.

[7] A. Ghosh, M. Mahdian, R. P. McAfee, and S. Vassilvitskii, "To match or not to match: Economics of cookie matching in online advertising," *ACM Transactions on Economics and Computation*, vol. 3, no. 2, p. 12, 2015.

[8] S. Englehardt, "The hidden perils of cookie syncing," *Freedom to Tinker*, 2014.

[9] J. M. Carrascosa, J. Mikians, R. Cuevas, V. Erramilli, and N. Laoutaris, "I always feel like somebody's watching me: Measuring online behavioural advertising," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, (New York, NY, USA), pp. 13:1–13:13, ACM, 2015.

[10] M. Stopczynski and M. Zugelder, "Reducing user tracking through automatic web site state isolations," in *International Conference on Information Security*, pp. 309–327, Springer, 2014.

[11] M. Falahrastegar, H. Haddadi, S. Uhlig, and R. Mortier, "Anatomy of the third-party web tracking ecosystem," *CoRR*, vol. abs/1409.1066, 2014.

[12] "The cost of ad blocking." http://downloads.pagefair.com/reports/2015_report-the_cost_of_ad_blocking.pdf, 2015.

[13] D. Thomas, "Ad blockers: How online publishing is fighting back." http://www.bbc.com/news/business-35602332. "[Online; accessed 20-May-2016]".

[14] H. Metwalley, S. Traverso, and M. Marco, "Unsupervised detection of web trackers," in *IEEE GLOBECOM*, 2015.

[15] T.-C. Li, H. Hang, M. Faloutsos, and P. Efstathopoulos, "Trackadvisor: Taking Back Browsing Privacy from Third-Party Trackers," in *PAM*, vol. 8995, pp. 277–289, 2015.

[16] F. Papaodyssefs, C. Iordanou, J. Blackburn, N. Laoutaris, and K. Papagiannaki, "Web identity translator: Behavioral advertising and identity privacy with wit," in *Proceedings of HotNets*, pp. 3:1–3:7, 2015.

[17] Z. Yu, S. Macbeth, K. Modi, and J. M. Pujol, "Tracking the trackers," Proceedings of WWW, pp. 121–132, 2016.

[18] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan, "Adscape: Harvesting and Analyzing Online Display Ads," in *WWW*, pp. 597–608, 2014.

[19] M. Falahrastegar, H. Haddadi, S. Uhlig, and R. Mortier, "The Rise of Panopticons: Examining Region-Specific Third-Party Web Tracking," in *TMA*, pp. 104–114, 2014.

[20] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, "Cookies That Give You Away: The Surveillance Implications of Web Tracking," in *WWW*, pp. 289–299, 2015.

[21] A. Barth, "HTTP State Management Mechanism," RFC 6265, 2011.

[22] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting on association for computational linguistics*, pp. 363–370, Association for Computational Linguistics, 2005.

[23] M. Ikram, H. J. Asghar, M. A. Kaafar, A. Mahanti, and B. Krishnamurthy, "Towards seamless tracking-free web: Improved detection of trackers via one-class learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 1, pp. 79–99, 2017.

[24] S. Sivakorn, I. Polakis, and A. D. Keromytis, "The cracked cookie jar: Http cookie hijacking and the exposure of private information," in *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 724–742, IEEE, 2016.

[25] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications," in *NDSS*, 2012.

[26] P. Eckersley, "How Unique is Your Web Browser?," in *PETS*, pp. 1–18, 2010.

[27] L. Olejnik, T. Minh-Dung, and C. Castelluccia, "Selling Off Privacy at Auction," in *NDSS*, pp. 199–208, 2013.